Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software UPM ETSIINF.
Exámen de Programación II. Convocatoria extraordinaria.08-07-2016.

Realización: *El test se realizará en la hoja de respuesta. Es* ***importante*** *que no olvidéis rellenar vuestros datos personales y el código clave de vuestro enunciado. Se pueden utilizar hojas aparte en sucio.*

Duración: *La duración total del test será de* ***40 minutos****.*

Puntuación: *El test se valora sobre* ***10 puntos****. Las preguntas tipo test pueden tener una única respuesta o varias respuestas, el enunciado lo deja claro. Cada pregunta con una única respuesta respondida correctamente vale 1 punto, e incorrectamente respondida resta 1/3 puntos. Si en una pregunta con una única respuesta se selecciona más de una respuesta, la pregunta se puntuará con 0 puntos. Para una pregunta con varias respuestas, cada afirmación correcta seleccionada suma* $1/no\_respuestas\_correctas$ *puntos, y cada afirmación incorrecta seleccionada resta* $1/no\_respuestas\_incorrectas$ *puntos. Las preguntas no contestadas suman 0 puntos en cualquier caso.*

Calificaciones: *Las calificaciones se publicarán en moodle como muy tarde el día* ***12 de julio de 2016***

Revisión: *Las revisiones serán el día* ***14 de julio de 2016*** *previa petición por correo electrónico al profesor que se indique por el foro de la asignatura.*

Primer Ejercicio

En la implementación del método *buscar* de la clase *ListaOrdenada* de la práctica 2:

```
/**
 * Posición del primer elemento con un valor
 * dado.
 * En el caso de no encontrarse retorna -1.
 * PRE: Cierto
 * POST: Retorna la posición del
 * elemento, o -1 si no se encuentra.
 * No se modifica actual
*/
public int buscar(String elemento)
```

Dada la siguiente implementación del método *buscar* que podría contener errores o no:

```
NodoTexto cursor = this.cabeza;
int posicion = 0;
while(cursor.getDato().compareTo(elemento)<0
        // No se ha llegado al final
        && cursor != null) {
        cursor = cursor.getSiguiente();
        posicion++;
}
if (cursor != null &&
  cursor.getDato().compareTo(elemento) == 0)
  return posicion;
else
  return -1;
```

Pregunta 1 (1 punto)

Dada la siguiente cadena: [“dos”, “cinco”, “cuatro”, “ocho”] Indique cuál sería el valor de posición justo después de salir del while si se busca **“siete”(sólo una)**.

a) Se produce un null pointer exception en la condición del while

b) Posición valdrá **4** porque se recorre toda la lista hasta que se determina que no está

c) Posición valdrá **-1** ya que siete no está en la lista

d) Posición valdrá **3** porque se recorre toda la lista hasta que se determina que no está

Segundo Ejercicio

Teniendo en cuenta el fragmento de código anterior

Pregunta 1 (1 punto)

Señale todas las afirmaciones verdaderas. **Puede haber más de una respuesta correcta**.

a) El código funciona si la lista está vacía

b) El código funciona si el elemento buscado se encuentra en la lista

c) El código funciona si el elemento no se encuentra en la lista

d) El código funciona si el elemento buscado no se encuentra en la lista pero este es menor que el último elemento de la lista

Tercer Ejercicio

Dado el siguiente código:

```
public class ElementoMovible {
 private int posX;
 private int posY;
 public ElementoMovible(){
 }
 public ElementoMovible (int x, int y){
  this.posX = x;
  this.posY = y;
 }
 /**
  * Desplaza el objeto x pixeles en horizontal
  * e y pixeles en vertical
  * @param x
  * @param y
  */
 public void mover (int x, int y){
  //.....
  posX+=x;
  posY+=y;
 }
}


public class Coche extends ElementoMovible {


 public Coche(){
```


Continúa...

```
 }

 public Coche (int x, int y)
 {
  super (x,y);
 }

 public void mover (int x, int y){
  //....
  super.mover(x+3, y-2);
 }
}

public class Ascensor extends ElementoMovible {

 public void mover (int y){
  //...
  super.mover(0, y);
 }
}
```

Pregunta 1 (1 punto)

Suponiendo que el código anterior compila correctamente, indique qué afirmación es la correcta con respecto al siguiente código (**sólo una**).

```
public static void main(String[] args) {
  ElementoMovible elemento =
    new ElementoMovible();
  Coche coche = new Coche();
  Ascensor ascensor = new Ascensor();
  ElementoMovible [] elemtos = {elemento,
      coche, ascensor};

  for (int i=0;i< elemtos.length;i++){
   elemtos[i].mover(i+3,i*2);
  }
 }
```

a) El código compila y se ejecuta correctamente ya que todas las clases proporcionan el método mover (int x, int y) bien porque la implementa la clase padre o bien porque la sobrescribe la clase hija

b) Se produce un error de ejecución porque en elemtos[2] hay una instancia de la clase Ascensor y esta clase no implementa el método mover (int x, int y);

c) Todas son falsas

d) Se produce un error de compilación porque en el array elemtos sólo puede haber objetos de la clase ElementoMovible

Cuarto Ejercicio

Dado el código del *Tercer Ejercicio*, suponiendo que compila correctamente, indique cuál sería el resultado del siguiente código

```
Coche coche2 = new Coche (1,2);
ElementoMovible element2 = coche2;
element2.mover(2, 0);
```

Pregunta 1 (1 punto)

Cuál de las siguientes afirmaciones es correcta (**sólo una**):

a) La nueva posición de coche2 no varía ya que mover se ejecuta en la clase ElementoMovible

b) La nueva posición de coche2 pasa a ser (2,0)

c) La nueva posición de coche2 pasa a ser (3,2)

d) La nueva posición de coche2 pasa a ser (6,0)

Quinto Ejercicio

Señalar las afirmaciones verdaderas. Puede haber más de una respuesta correcta

Pregunta 1 (1 punto)

Puede haber más de una respuesta correcta.

a) Java permite herencia múltiple de clases (una clase hija puede tener varias clases padre)

b) Una clase padre solo hereda los métodos públicos de sus clases hijas.

c) Una clase padre puede tener más de una clase hija.

d) Todos los objetos (instancias) de una clase hija son objetos también de la clase padre.

Continúa...

Sexto Ejercicio

Dado el siguiente código y supuesta la correcta definición de las excepciones `ErrorLocalizadorNoEncontrado` y `ErrorLocalizadorUsadoDado`:

```
public class TratarLocalizador {
 public String buscarPuerta(long nLoc)
 throws ErrorLocalizadorNoEncontrado, ErrorLocalizadorUsado{
  if (nLoc%150 == 0)
   throw new ErrorLocalizadorNoEncontrado();
  if (nLoc < 3000 && nLoc >150){//IF
   return "GA"+(int)nLoc/10;
  }//IF
  else
  {
   throw new ErrorLocalizadorUsado();
  }
 }
}

public static void main(String[] args) {//main
 long [] localizadores = {240,150,3001,347};
 TratarLocalizador tratar = new TratarLocalizador();
 int i=0;
 try{//Try-catch1
  for (i=0;i<localizadores.length;i++){//FOR
   String puerta = tratar.buscarPuerta(localizadores[i]);
   System.out.println(localizadores[i]+" embarque por: "+puerta);
  }//FOR
 } catch (ErrorLocalizadorNoEncontrado e) {
  System.out.println("No se encuentra puerta para el localizador: "+localizadores[i]);
 } catch (ErrorLocalizadorUsado e) {
  System.out.println("El localizador: "+localizadores[i]+" ya ha sido usado");
 }//try-catch1
}//main
```

Pregunta 1 (1 punto)

Indique cuál de las siguientes afirmaciones es cierta (**sólo una**):

a) El sistema indica la puerta del localizador 240, indica que para el localizador 150 no se encuentra puerta de embarque y termina la ejecución cuando se intenta procesar el localizador 3001 indicando que el localizador ya ha sido usado

b) Se procesan todos los localizadores del array localizadores, pero no se muestra la puerta de embarque del localizador 150, porque se produce la excepción ErrorLocalizadorNoEncontrado

c) Se procesan todos los localizadores del array localizadores mostrando la puerta de embarque para cada uno

d) Sólo se procesa el localizador 240, porque al procesar el 150, se produce una excepción y la aplicación termina

Continúa...

Séptimo Ejercicio

Dada la siguiente definición de clase:

```
public class Operaciones {
 private int datoEntero = 0;
 private double datoReal = 0.0;

 public Operaciones (){

 }
 public Operaciones (int datoEntero){
  this.datoEntero = datoEntero;
 }

 public Operaciones (double datoReal){
  this.datoReal = datoReal;
 }

 public Operaciones (int datoEntero, double datoReal){
  this.datoEntero = datoEntero;
  this.datoReal = datoReal;
 }

 public void computa (int dato){
  datoEntero = datoEntero*dato+(int)(datoReal/dato);
 }

 public void computa (double dato){
  datoReal = datoReal-dato+datoEntero*dato;
 }
}
```

Pregunta 1 (1 punto)

Indique qué afirmación es correcta (**sólo una**):

a) Para que compile el código hay que cambiar el nombre del parámetro de uno de los métodos 'computa', por ejemplo, de dato a dato2, de esta forma sí que se puede aplicar la sobrecarga

b) El código compila sin problemas gracias a la sobrecarga

c) Este código no compila porque hay dos implementaciones del método computa y Java no sabría cuál usar

d) Este código no compila porque tiene varios constructores

Octavo Ejercicio

Pregunta 1 (1 punto)

Se llama constructor copia de una clase a (**sólo una**):

a) Un método constructor de una clase que recibe una instancia de la propia clase y crea un nuevo objeto copia del proporcionado como parámetro

b) Un método que no recibe nada y devuelve un objeto copiado del objeto sobre el que se invoca el método (this)

c) Un constructor de una clase llamada *Copia*

d) En programación orientada a objetos no existe el concepto de constructor copia

---

Continúa...

Noveno Ejercicio

Dado la siguiente definición de las clases *Persona*, *Pareja* y *TestPareja*:

```
public class Persona {
 private String nombre;
 public Persona (String nombre){
  this.nombre = nombre;
 }
 public void setNombre(String nombre){
  this.nombre = nombre;
 }
 public String toString(){
  return nombre;
 }
 public boolean esIgual(Persona p){
  return nombre.equals(p.nombre);
  }
}
public class Pareja {
 private Persona personaA;
 private Persona personaB;
 public Pareja (Persona personaA, Persona personaB){
  this.personaA= personaA;
  this.personaB= personaB;
 }
 public void getPersonaA (){
  return personaA;
 }
 public void getPersonaB (){
  return personaB;
 }

public String toString(){
  return "par [" +personaA+ ", " +personaB+ "]";
 }
}

public class TestPareja {

 public static void main (String[] args){
  Persona persona1 = new Persona("Juan Perez");
  Pareja par = new Pareja(persona1, new Persona("Luis Sol"));
  persona1.setNombre("Pedro");
  System.out.println(par);
 }
}
```

Pregunta 1 (1 punto)

Indique qué salida por consola se producirá (**sólo una**):

a) par [Juan Perez, Luis Sol]

b) par [Pedro, Luis Sol]

c) par [Luis Sol, Luis Sol]

d) Pareja@02AF1239

Décimo Ejercicio

Dado el código del ejercicio anterior y el siguiente código de *TestPareja2*:

```
public class TestPareja2 {

public static void main (String[] args){
 Persona persona1 = new Persona("Juan Perez");
 Persona persona2 = new Persona("Juan Perez");
 Persona persona3 = persona2;
 System.out.print(persona1 == persona2);
 System.out.print(persona2 == persona3);
 System.out.print(persona1.esIgual(persona2));
 System.out.print(persona2.esIgual(persona3));
 }
}
```

Pregunta 1 (1 punto)

Indique qué salida por consola se producirá (**sólo una**):

a) false true true false

b) true true true true

c) false true true true

d) false false true false

Continúa...